FIX Reference Manual 1.0
By Michael Ekberg, 1/26/81

I. INTRODUCTION TO FIX

Fix is an assembly language utility to examine and modify disk files. All numbers outputted by and entered to FIX are hexidecimal. The ctrl 1 key can be used to start and stop the data display on the screen.

II. FILE STRUCTURE

Diskettes are divided into blocks of data called sectors. The ATARI 810 has 720 sectors of 128 bytes. The ATARI 815 has 720 sectors of 256 bytes. Diskettes have four parts:

1. BIT MAP. The bit map contains information on the status of each sector. By examining the Bit Map the File Manager can determine if a sector has been allocated(in use) or not(free).
2. DIRECTORY. The directory sectors contain information on the names, location, length, and status of the files on a diskette.
3. BOOT SECTORS. Each diskette contains 1 or 3 sectors reserved for information read on power-up. These sectors are normally inaccessable to the user when using the File Manager. All 9/24/79 DOS diskettes use sectors 1 for the boot. All DOS II 2S or 2D diskettes use sectors 1,2 and 3.
4. DATA SECTORS. The rest of the sectors are used to store the information in the files. Most of the bytes of a sector are dedicated to the actual file information. The last three bytes of every sector contains "house-keeping" information for the File Manager.

FILES are made up of a DIRECTORY ENTRY and a set of one or more sectors. The sectors are usually contiguous but do not have to be. For a complete description of a directory entry see section III.A

III. MENU ITEMS

FIX is a menu driven program. To execute commands, type the letter in front of the command listed in the displayed menu(see Fig. 1).
You do not need to type <RETURN>. FIX automatically accepts your command and replies with the prompt for that command. If you have inadvertantly typed the wrong command, hit the <BREAK> key to abort the command.

A: Directory Entries.

This menu item will display the entries in the directory. Fig.1 shows the user requesting FIX to display the first 8(hex) entries. Notice that the entries start with 0. The directory may contain a maximum of 64 files. Single entries may be displayed by simply typing the file number and <RETURN>.

The left most column gives the file number of each file displayed. The next column is the filename and extension. Notice that the period is omitted in the file name. The next column, labelled FSEC, is the first sector in the file. The #SEC column gives the number of sectors in the file. Both numbers are hexadecimal.

The last column, DL, gives the status of the file. Notice in Fig. 2 that file 4(FIX6) has a D. This means that the file has been deleted from the directory. Since it the first available "hole" the next file created will occupy its directory entry.

File 3(MYPROG.BAS) has an L. The L indicates that the file is locked. Also notice that files 5-8 are empty directory entries, containing no directory entries.

B: Trace Sector Chain

This menu item will verify the structure of a file. It traces the file through all of its sectors until it hits end of file or a bad link. A link is a pointer to the next sector of a file. A bad link is a link to a sector that belongs to another file. An end of file is a link that points to the sector 0.

Fig.3 shows the sector chain for the small BASIC file MYPROG.BAS. Notice that we are tracing file number 3. The file number is the entry number into the directory.

The BS column has two different meanings, depending on the DOS that created the file. The 9/24 DOS A used this number as the relative sector number. For the first sector in a file this should be 1. Subsequent sectors should add 1 to this number. If you notice that numbers are skipped, this may indicate that the links in your file may have become bad.

The last sectors BS is the number of bytes in the sector.

For DOS II the BS indicates the number of bytes in each sector. Most sectors should contain $7D bytes(128 byte single density disk) or $FD bytes(256 bytes double density disk). The last sector in a file is usually less then the above numbers.

Appending one file to another will cause the byte count in the sector inbetween the appended portions to be less then full . It is therefore possible to have partially filled sectors in the middle of a file .

The last column, FP, is the forward pointer. This is the link to the next sector of the file. Files on newly formatted diskettes tend to be linked sequentially. Files on diskettes that have several files that have been deleted tend to become "fractured". This means that the files tend not to be sequentially linked. In other words the sectors are not physically contiguous.

C: Modify Directory Entry

You can use this menu item to make changes to the diskettes directory. Enter the file number of the entry you wish to change followed by <RETURN>.

Fig. 4 shows the user is making a change to directory entry 3. FIX has printed the directory item and positioned the cursor to the filename.

At this point you can edit the filename, the starting sector number, the number of sectors, or the status of the file. By inserting or deleting the D you can delete or "un-delete" the file. By adding or deleting the L you can lock or unlock the file. See section III.A for the meaning of the rest of the info on the line.

If you wish to have FIX update the entry, then type <RETURN>. If you don't want the entry updated, then type <BREAK>.

D: Check Allocation Map

This menu item can recover sectors that have become mis-allocated. To run it type D. FIX automatically starts making it's own BIT MAP. It will link thru all the files and mark all the sectors in use and those that are free. FIX then compares it's BIT MAP to the BIT MAP on the diskette. All sectors that should be free but aren't are printed as "MARKED IN USE". All sectors that are really allocated to a file but aren't in the diskettes BIT MAP are marked "FREE". Sectors that are correctly allocated are not printed.

After FIX has checked all the sectors it will exit back to the menu if all sectors are correct. If some sectors are mis-allocated it will ask you to if you want to write it's correct copy to the diskette. Just type "Y" to do so. If you don't want to write the corrected BIT MAP to be written to the disk then type <RETURN>.

Fig. 5 shows the result of a D command. Notice that we had a bad link in file 0(the first file). This will usually cause an error 164 when you attempt to read the file. We would probably want to trace its sector chain (B command) to find the bad link.

Our bit map seems to be slightly mangled also, so we would type Y to rewrite our new bit map.

E: Modify Sector Link

You may have discovered that some of your sector links have been altered incorrectly. This menu item allows you to change the links in the sectors on the diskette. After you type E , FIX will prompt you to enter the sector number to modify. Enter the hex sector number followed by <RETURN>

Fix then prints the sector link information from the sector on the diskette. FIX positions the cursor on the line. Using the cursor control keys, you can alter the file number this sector is allocated to, the number of bytes in this sector of the file, and the FORWARD POINTER. Typing an E to the right of the FORWARD POINTER will mark this sector as End of File.

To tell FIX to write out the updated info to the sector simply type <RETURN>. If you realize that you have made an error in trying to modify this sector, position the cursor below the line and type <RETURN>. FIX will then re-display the line. Type <RETURN> to exit.

See section III.B for more info on the sector link display.

F: Set Drive Number

This menu item sets the drive number. Type the drive number (1-8) and <RETURN>. Fig. 7 shows a user setting the drive number to 1 (FIX defaults to drive 1 upon loading).

G: Exit To Dos

This item causes a return to DOS.

IV. ERROR CODES AND RECOVERY

ERROR #	DESCRIPTION

128 BREAK KEY ABORT
This is caused by typing <BREAK>.

129 IOCB ALREADY OPEN
The IOCB is already open.
Recovery: You can CLOSE the IOCB and re-OPEN it. You can use the current IOCB. You may want to check and see why the IOCB is OPEN.

130 NON EXISTENT DEVICE
You have tried to access a device not in the handler table, ie. the device is undefined. The handler may be loaded in memory but not initialized. This error may occur when trying to access the ATARI 850 without running the RS232 AUTORUN.SYS file.
Recover: Check your I/O command for the correct device. Or load and initialize the correct handler.

131 IOCB WRITE ONLY ERROR
You have attempted to read from a file opened for write only.
Recovery: Open the file for read or read/write (update mode).

132 ILLEGAL HANDLER COMMAND
This is an CIO error code. The command code passed to the device handler is illegal. The command is either <=2 or is a special command to a handler that hasn't implemented any special commands.
Recovery: Check your XIO, or IOCB command code for illegal command code.

133 DEVICE/FILE NOT OPEN
Device or file not open. You have not OPENED this file or device.
Recovery: Check your OPEN statement or file I/O statement for the wrong file specification.

34 INVALID IOCB NUMBER
You have tried to use an illegal IOCB index. For BASIC, the range is 1-7. BASIC does not allow use of IOCB 0. The ASSEMBLER EDITOR cartridge requires the IOCB index to be a multiple of 16 and less than 128.

135 IOCB WRITE ONLY ERROR
You have tried to write to a device/file that is OPEN for read only.
Recovery: You could open the file for write or read/write.

136 END OF FILE
Your input file is at end of file. No more data in file.

137 TRUNCATED RECORD
This error typically occurs when the record you are reading is larger then the maximum record size specified in the call to CIO. BASIC's maximum record size is 119 bytes.
Recovery: You probably are trying to use an INPUT(Record-oriented) type command on a file that was created using PUT(byte-oriented) commands.

Records in a file are delimited by EOL(End Of Line) characters ($9B). Files that are created using PUTs have no EOL unless the bytes you output have a $9B. Trying to read a PUT file with record I/O may cause the DOS to read a record that is the size of the file. The DOS will then generate this error. Try reading the file using PUT type statements.

This error may also occur when ENTERing a BASIC program created using SAVE. Try LOADing the program.

138 DEVICE TIMEOUT
A command was sent by the computer to a device device over the SERIAL BUS. The device did not respond within the period set by the O. S. for that particular device command.

This error can be caused in several ways.
1. The device number may be wrong.

Ex. OPEN #3,4,0,"D3:MYPROG"

You would get this error if disk drive 3 was not connected to the computer, not turned on, or not present.
2. The disk drive may be present but is unable to execute

the command in the proper period of time.
Recovery: To recover from error case 1, examine the connections between the disk drive and the computer to make sure the are properly secured. Check the drive to make sure it is powered on and set for the correct drive number. You should check your I/O command for the correct drive number. Retry the command. If you get this error again see the next line.

The second type of timeout error is caused by an interaction between the O. S. and the disk drive. DOS 9/24's timeout value was set too low for all possible cases of disk operations. The disk may have had some trouble executing the commmand in the short period allowed by the operating system. DOS II does not have this problem.

The second type of timeout error is intermitent. You should be able to retry the operation and succeed. If this error occurs more then a few times, the device may need repair.

139 DEVICE NAK
This error is a sort of catch all error code. The device may have received a valid command but can't execute it because of bad parameters. For example, trying to read an unaddressable sector, such as sector 0. The device may have recieved a garbled command or data frame from the computer.
Recovery: Check your I/O command for illegal parameters.

140 SERIAL FRAME ERROR
Bit 7 of SKSTAT in POKEY is set. This means communications from the device to the computer is garbled. Specifically, POKEY detected missing or extra bits in a byte received on the SERIAL DATA BUS.

141 CURSOR OUTRANGE
Your cursor is out of range for this particular graphics mode.

142 SERIAL OVERRUN
Bit 5 of SKSTAT in POKEY is set.The computer did not respond fast enough to a SERIAL BUS input interupt. POKEY recieved another 8-bit word on the SERIAL BUS before the computer could process the previous word received.
Recovery: This is a rare error. If it recurs you should have your computer serviced.

143 CHECKSUM ERROR
Serial Bus communications are screwed up. The checksum sent by device is not the same as that calculated for frame received by computer.
Recovery: Not much you can do about this. Could be hardware or sortware problem in device/computer.

144 DEVICE DONE ERROR
The device is unable to execute a valid command. There are two causes of this error. Usually it means you have tried to write to a write protected device or diskette. This is easily corrected by removing the disk protect tab or turning the write protect switch on the 815 off.

The second reason for this error appears unknown at this point.The disk drive is unable to read/write the sector requested or the cassette is set at the wrong baud rate.
Recovery: Remove write protect tab or turn off write protect switch. If the disk was not write protected, you may have problems with the diskette media. Don't know how to recover on cassette.

145 READ AFTER WRITE COMPARE ERROR
You have tried to open the SCREEN EDITOR with an illegal graphics mode number.
Recovery: Check GRAPHICS mode call or the AUX2 byte in the IOCB.

146 FUNCTION NOT IMPLEMENTED
Function not implemented in handler, ie. trying to PUT to the keyboard or issuing special commands to the keyboard.
Recovery: Check I/O command for right command to correct device

147 NOT ENOUGH RAM FOR SELECTED GRAPHICS MODE
Insufficient RAM for graphics mode selected.
Recovery: Add more memory or use a smaller graphics mode.

160 DRIVE NUMBER ERROR
You specified an out of range drive(not 1-8) or you have not allocated a buffer for this drive.
Recovery: Check your file specification or byte 1802 ($710).

161 TOO MANY OPEN FILES
You don't have any free sector buffers to use on another

file.
Recovery: You may have up to 8 open files. Check location 1801 ($709) for the number of sector buffers allocated.

162 DISK FULL
You don't have any more free sectors on this diskette.
Occurs when writing to a full diskette.
Recovery: Use a different diskette with free sectors. Use the D option on FIX to see if you have any mis-allocated sectors. At present there is no way to recover from this error during program execution.

163 FATAL SYSTEM I/O ERROR
This error code means that the file manager has a bug in it. If you get this error then please report it to CUSTOMER SERVICE.

164 FILE NUMBER MISMATCH
The structure of the file is damaged. One of the file links points to a sector allocated to another file.
Recovery: Use B and E FIX commands to fix up the file links. You can not recover from this error during program execution.

165 FILENAME ERROR
Your file specification has illegal characters in it. Legal characters are alphnumerics,*, and ?.
Recovery: Check file specification and remove illegal characters.

166 POINT DATA LENGTH
The byte count in the point call was greater then 125 (single density) or 253 (double density
Recovery: Check POINT statement parameters.

167 FILE LOCKED
You tried to access a locked file.
Recovery: Unlock file.

168 DEVICE COMMAND INVALID
Probably means device didn't recognize command.
Recovery: ?????

169 DIRECTORY FULL

You don't have any free entries in the directory.
Recovery: See A and C FIX commands.

170 FILE NOT FOUND
You have tried to access a file that doesn't exist in the diskettes directory.
Recovery: Usually you have mistyped the file spec. Look at the directory using the DOS A command for the correct name.

171 POINT INVALID
You've probably tried to POINT to a byte or record beyond the end of the file.
Recovery: Check size of file against the POINT value.

172 ILLEGAL APPEND
You have tried to OPEN a DOS I file for append using DOS II. DOS II cannot append to DOS I files.
Recovery: Copy the DOS I file to DOS II diskette using DOS II.

173 BAD SECTORS AT FORMAT
The disk controller detected bad sectors while FORMAT'ing a diskette.
Recovery: Throw away the diskette and use another. If you cannot FORMAT a diskette, the disk drive may need repair.

```
FIXDUMP 25B (C) ATARI INC 9/11/1980

A: DIRECTORY ENTRIES
B: TRACE SECTOR CHAIN
C: MODIFY DIRECTORY ENTRY
D: CHECK ALLOCATION MAP
E: MODIFY SECTOR LINK
F: SET DRIVE NUMBER
G: EXIT TO DOS



SELECT ITEM OR RETURN FOR MENU
```

Fig. 1

```
SELECT ITEM OR RETURN FOR MENU
FIRST,LAST DIR ENTRIES TO SHOW?
0,8
DN FILENAM EXT FSEC NSEC DL
00 DOS     SYS 0004 0026
01 DUP     SYS 002A 002A
02 FIX6        0054 002A
03 MYPROG  BAS 007E 000I  L
04 FIX6        0054 0028 D
05 ********** 0000 0000
06 ********** 0000 0000
07 ********** 0000 0000
08 ********** 0000 0000

SELECT ITEM OR RETURN FOR MENU
```

Fig. 2

```
SELECT ITEM OR RETURN FOR MENU
TRACE FROM WHAT DIR ENTRY?
3
01 MYPROG   BAS 007E 0001   L
SECTOR 07E: FN=01, DS=7D, FP=07F
SECTOR 07F: FN=01, DS=7D, FP=000
SECTOR 000: FN=07, DS=45, FP=000 E

SELECT ITEM OR RETURN FOR MENU
```

Fig. 3

```
SELECT ITEM OR RETURN FOR MENU
WHICH ENTRY TO MODIFY?
3
DR FILENAM EXT FSEC NSEC DL
01 MYPROG   BAS 007E 0001   L
```

Fig. 4

```
SELECT ITEM OR RETURN FOR MENU
BUILDING ALLOCATION MAP...
BAD LINK IN FILE # 00
SECTOR 001: FN=00, DS=0C, FP=201
WAS MARKED IN-USE
SECTOR 004: FN=00, DS=7D, FP=005
WAS MARKED FREE
SECTOR 007: FN=00, DS=7D, FP=000
WAS MARKED IN USE
SECTOR 02A: FN=01, DS=7D, FP=02D
WAS MARKED FREE
SECTOR 02C: FN=01, DS=7D, FP=02D
WAS MARKED FREE
SECTOR 032: FN=01, DS=7D, FP=033
WAS MARKED FREE
```

Fig. 5

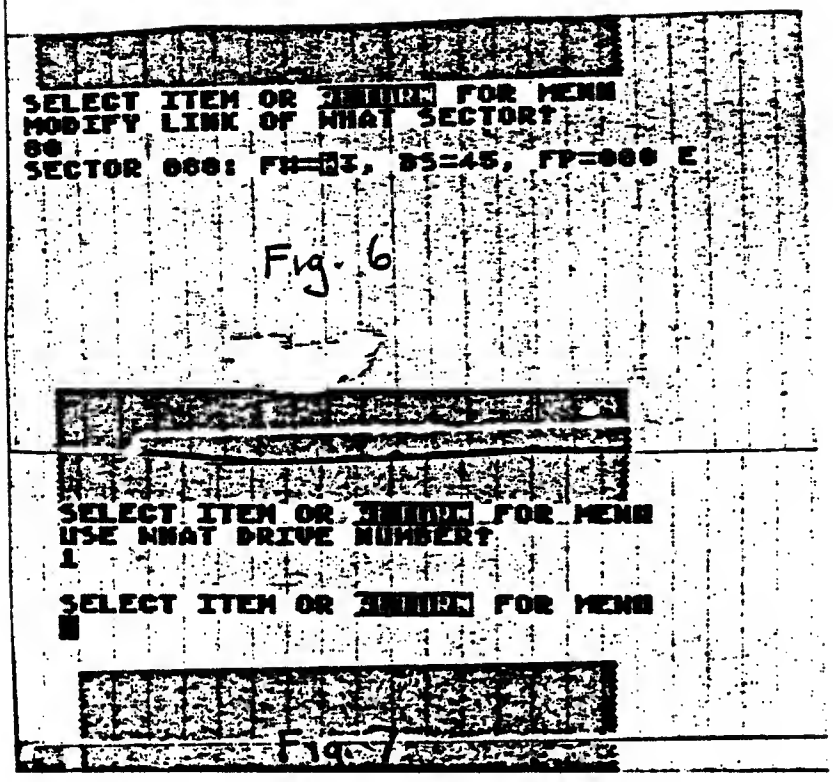
SELECT ITEM OR [illegible] FOR MENU
MODIFY LINK OF WHAT SECTOR?
00
SECTOR 000: FW=03, BS=45, FP=000 E

Fig. 6

SELECT ITEM OR [illegible] FOR MENU
USE WHAT DRIVE NUMBER?
1
SELECT ITEM OR [illegible] FOR MENU

Fig. 7